예측주기를 이용한 트랜스코딩 기반의 스트리밍 시스템

(A Streaming System based on Transcoding using the Prediction Period)

김 성 민 † 김 현 희 †† 박 시 용 ††† 정 기 동 ††††
(Sung Min Kim) (Hyun Hee Kim) (Si Yong Park) (Ki Dong Chung)

요 약 멀티미디어 서비스가 전체 인터넷의 많은 부분을 차지하고 있으며, 계속해서 관심이 증가되고 있다. 네트워크가 다양해지고 구성 단말들의 종류도 다양해짐에 따라, 획일적인 컨텐츠로 멀티미디어 서비스를 하는 것이 불가능해졌다. 각 단말별 요구도 다양해지고, 네트워크의 대역폭도 다양해짐에 따라서 적응적인 서비스가 요구된다. 비디오 트랜스코딩은 멀티미디어 데이타를 적응성 있게 서비스할 수 있는 좋은 방법이다. 본 논문은 멀티미디어 데이타를 스트리밍 하기 위해 부호기, 트랜스코더, 복호기로 구성된 시스템을 제안한다. 부호기는 예측 주기를 이용하여 트랜스코딩할 경우의 계산 복잡도 및 PSNR을 향상시켰고, 복호기는 일반적인 미디어 재생기와 거의 유사하다. 트랜스코더에서는 부호기의 예측주기와 프레임율을 조절하는 삭제주기를 통해서 효율적인 계산 과정을 선택한다. 실험 결과, 기존의 트랜스코더와 부호기, 복호기를 이용한 시스템에 비해서 본 논문에서 제안한 시스템이 계산복잡도와 PSNR에서 높게 나타났다.

키워드 : 트랜스코딩, 스트리밍 시스템, 예측 주기, 계산 복잡도

Abstract Multimedia is a very popular service in the Internet. But, we cannot provide multimedia service at a uniform content, because networks and devices are various. Thus, an adaptive service is needed for multimedia transmission. Video Transcoding is the good solution that can service multimedia adaptively. This paper proposes the streaming system that is composed of encoder, transcoder, decoder. The encoder enhanced time complexity and PSNR in case of transcoding using PP(Prediction Period). The decoder is almost same as the traditional media player. Transcoder reduced time complexity through combination of prediction period in encoder and skipping period to control frame rate in transcoder. In simulation results, the performances of proposed scheme outperform the system with traditional transcoder in time complexity and PSNR.

Key words : Transcoding, Streaming system, Prediction period, Time complexity

1. 서 론

멀티미디어 응용은 우리의 일상에서 큰 비중을 차지하고 있다. 오늘날의 멀티미디어 기술은 네트워크 서비스 제공자가 홈쇼핑, 게임, 그리고 주문형 비디오와 같은 다양한 서비스를 제공할 수 있도록 지원한다[1,2]. 이러한 응용에서 비디오 스트리밍 기술은 미디어 전송에서 중요한 역할을 담당한다. 비디오 스트리밍을 통해 다양한 서비스가 가능하고 상업적으로도 잠재성을 가지고 있으므로 많은 회사, 기관, 그리고 대학에서 제품뿐만 아니라 표준과 새로운 기술을 개발하고 있다[3-7].

네트워크와 장치의 종류, 그리고 컨텐츠를 표현하는 포맷의 수가 다양해짐에 따라 이질적인 시스템과 네트워크 간의 상호 운용성이 더욱 중요하게 되었다. 비디오 컨텐츠의 트랜스코딩은 이것을 가능하게 만들어주는 중요한 기술이다. 그림 1은 비디오 트랜스코더를 이용한 스트리밍 서비스를 나타내고 있다. 비디오 트랜스코딩은 임의의 포맷, 크기, 그리고 비트율로 부호화된 입력 비디오 스트림을 이질적인 네트워크 환경의 클라이언트에게 효과적으로 서비스하기 위하여 다른 속성의 비디오

† 학생회원 : 부산대학교 컴퓨터공학과
morethannow@pusan.ac.kr
†† 정 회 원 : 동양종합금융증권(주) 컨텐츠팀
h2k@melon.cs.pusan.ac.kr
††† 정 회 원 : 대전대학교 교육개발센터 전임강사
sypark@dju.ac.kr
†††† 종신회원 : 부산대학교 컴퓨터공학과 교수
kdchung@melon.cs.pusan.ac.kr
논문접수 : 2005년 2월 21일
심사완료 : 2006년 9월 5일

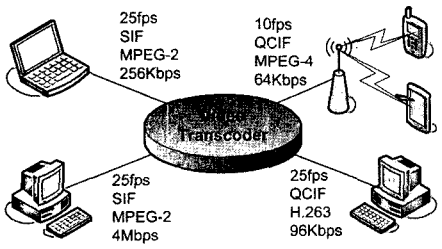


그림 1 비디오 트랜스코더를 이용한 스트리밍 서비스

스트림으로 변환하는 기술이다[9-11]. 부호화된 입력 스트림의 속성을 변경하기 위해서는 복호화와 재부호화 과정이 동시에 필요하므로, 비디오 트랜스코더는 그림 2와 같이 복호기와 부호기로 구성된다.

이질적인 네트워크 환경의 클라이언트를 지원하기 위한 대부분의 트랜스코딩 기법들은 비디오 스트리밍 시스템에서 트랜스코더 내부에만 한정되어 있다. 입력된 비디오 스트림의 화질 열화는 인간의 시각 시스템이 허용하는 범위 내로 쉽게 조절할 수 있지만, 계산 복잡도의 최소화에는 한계가 있다. 본 논문에서는 서버의 부호기에서 기존의 비디오 스트림의 구조를 변경하여 트랜스코더가 간단하게 프레임율을 조절할 수 있는 경량화된 스트리밍 시스템을 제안한다.

본 논문에서 제안한 서버의 부호기는 예측 주기를 통해 기존의 비디오 구조를 변경하고, 서비스되는 컨텐츠를 트랜스코딩 할 경우에 처리량을 줄일 수 있도록 하였다. 트랜스코더에서는 예측 주기를 통해서 서비스되는 컨텐츠를 변환하기 위해서 삭제 주기를 이용한다. 예측 주기를 통해서 서비스되는 컨텐츠는 예측 주기에 따라서 바로 직전에 있는 예측 주기 이전의 프레임들을 참조하지 않으므로, 예측 주기와 삭제 주기를 통해서 변환을 위한 처리가 필요한 프레임과 그렇지 않은 프레임으로 구분할 수 있다. 변환 처리가 필요하지 않은 프레임들은 단순히 삭제가 가능한 프레임들로, 프레임율을 변환할 경우 기존의 트랜스코더에 비해서 계산복잡도를 줄일 수 있다. 복호기에서는 예측 주기를 이용하는 부호기에서 서비스되는 컨텐츠가 트랜스코딩이 필요하지 않는 클라이언트에서 올바로 재생되지 못할 수 있으므로 트랜스코딩 여부에 따라 처리를 달리 할 수 있도록 제안한다. 또한 본 논문에서는 프레임율 조절을 위하여 프레임을 제거할 때 발생되는 시간적 연관성 손실을 해결하기 위한 개선된 움직임 예측 방법으로 움직임 벡터 합성 기법을 제공하여 화질 열화나 에러 축적 현상을 감소시켰다.

실험을 통한 결과에서 제안된 스트리밍 시스템은 기존의 방법에 비해 비슷한 출력 화질과 약 8%~65%정도의 계산 복잡도로 향상된 성능을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 트랜스코더 구조와 트랜스코딩 알고리즘에 관한 연구들에 대해 살펴본다. 3장에서는 제안한 서버 부호기의 알고리즘과 움직임 벡터 합성 알고리즘을 설명하고, 제안한 비디오 스트리밍 시스템의 성능을 평가하고 분석한다. 4장은 본 논문의 연구 내용을 요약하고 향후 연구 방향을 제시한다.

2. 관련 연구

트랜스코딩에 관한 대부분의 연구는 트랜스코더의 구조와 움직임 예측 알고리즘을 중심으로 수행되었다[8]. 이는 트랜스코더 내부 처리 과정에 따라 계산 복잡도와 비디오 화질의 편차가 다를 수 있고, 개선된 움직임 재사용을 통해 화질 및 계산 복잡도에서 성능을 향상시킬 수 있기 때문이다.

비디오 트랜스코딩은 입·출력 스트림의 부호화 규격을 기준으로 크게 동기종 변환과 이기종 변환의 두 범주로 나눌 수 있다[12]. 동기종 비디오 트랜스 코딩은 출력 스트림의 부호화 규격을 입력 스트림과 동일하게 유지하며 양자화 인자, 프레임율, 해상도를 변환하는 기법이고, 이기종 비디오 트랜스 코딩은 출력 스트림의 부호화 규격 및 속성을 입력과 다르게 변환한다.

동기종 비디오 트랜스코딩에서 양자화 인자 변환 방법은 복호화 후의 부호화 과정에서 적절한 양자화 인자를 선택하는 간단한 방법으로 비트율을 조절한다[13-16]. 그러나 트랜스코더 내부 부호기의 재양자화로 인해 화질 열화가 발생하는 단점이 있다. 프레임율 변환 방법은 입력 스트림의 프레임을 반복적으로 제거하여 비트율을 조절한다[17-19]. 하지만 비트율을 줄이기 위해 단순하게 프레임을 제거하면 프레임 간의 시간적 연관성도 손실되어 화질 열화나 에러 축적 현상이 발생하는 단점을 가진다. 해상도를 변환하는 기법은 Mobile Phone, PDA와 같은 비디오 재생 크기에 제약을 가진

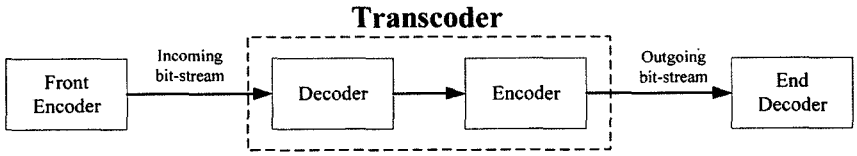


그림 2 비디오 트랜스코더의 구조

단말기를 지원하기 위해 입력된 스트림의 화면 크기를 조절하는 데 사용된다[20].

프레임율 조절 트랜스코더에서는 제거된 프레임에 따른 화질 열화를 막기 위해, 제거되지 않은 프레임 간에 새로운 시간적 연관성을 얻는 것이 필요하다. 움직임 재예측 기법은 제거되지 않은 프레임 간의 움직임 벡터를 재계산하므로 높은 정확성에 비해 많은 계산 복잡도를 요구한다[21]. 따라서 제거된 프레임의 움직임 벡터를 재사용하여 프레임 간의 시간적 연관성을 얻을 수 있는 움직임 벡터 재사용 기법에 대해 많이 연구되었다[18,19,22].

2.1 CPDT(Cascaded Pixel-Domain Transcoder)

비트율 감소의 목적은 가능한 계산 복잡도를 낮게 유지하면서 높은 화질을 얻는 것이다. 비트율이 감소된 스트림은 비트율만 감소되고 화질 저하는 없어야 이상적이라 할 수 있다. 이것을 위한 가장 일반적이고 직접적인 방법은 그림 3에 나타나 있는 것처럼 입력된 비디오 스트림을 완전히 복호화하여 새로운 비트율로 재부호화하는 것이다. 그러나 이 기법은 높은 화질을 보장하는 반면 높은 계산 복잡도를 요구하므로 실시간 비디오 스트리밍 서비스에 적용될 수 없는 단점을 가진다.

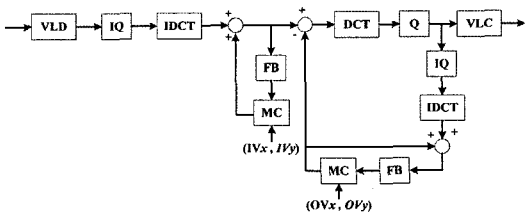


그림 3 픽셀 영역 기반의 트랜스코더

2.2 DDT(DCT-Domain Transcoder)

비트율 감소를 위해서 부호화와 복호화 과정은 반드시 필요하지만 너무 높은 계산 복잡도는 실시간 스트리밍 서비스에 장애가 된다. 비디오 트랜스코딩에서 움직임 벡터 예측 과정에 많은 시간이 소모되므로 계산 복잡도를 줄이기 위하여 제안된 구조가 DCT-Domain Transcoder이고 그림 4에 나타나 있다. CPDT에 비해 간단한 구조이고 낮은 계산 복잡도로 많은 연구에 사용되고 있다. 그러나 움직임 예측 과정의 부재는 복호화된 스트림과 부분적으로 재생성되어 부호화된 스트림 간의 불일치를 초래하여 에러 축적 현상을 발생시킨다.

2.3 Dynamic Frame Skipping Transcoder

[17]에서는 양자화 에러나 높은 계산 복잡도를 개선하기 위하여 매크로블록의 모드에 기반한 트랜스코더 구조를 제안하였고, 우수한 성능을 가지고 있다. 스위치로 매크로블록의 모드를 식별하고 해당 모드가 MC 또는

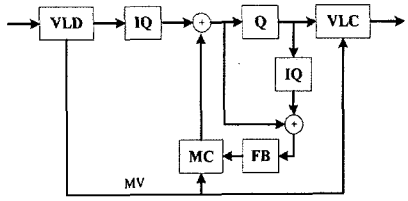


그림 4 DCT 영역 기반의 트랜스코더

Non-MC인지에 따라 트랜스코딩 기법을 달리 적용할 수 있어서 적은 계산 복잡도와 높은 화질을 얻을 수 있을 뿐만 아니라 클라이언트의 가용 대역폭에 따라 비디오 스트림의 프레임율을 동적으로 조절할 수 있는 구조이다. MC 모드로 부호화된 매크로블록은 움직임 벡터가 지정하는 영역이 이전 프레임 내에 있는 임의의 여러 매크로블록들에 걸쳐져 있지만 Non-MC 모드의 매크로블록은 움직임 벡터가 이전 프레임 내에 있는 임의의 하나의 매크로블록과 정합된다.

그림 5에는 매크로블록의 모드에 기반하여 동적인 프레임율 조절이 가능한 트랜스코더가 나타나 있다. 비디오 스트림이 입력되면 먼저 VLD(Variable Length Decoder)는 매크로블록의 헤더 정보, 부호화 모드, 움직임 벡터, 그리고 양자화된 DCT 계수를 추출한다. 양자화된 DCT 계수는 역양자화기(Inverse Quantizer)를 거치면서 DCT 계수로, 역DCT(Inverse Discrete Cosine Transform)를 통해 픽셀 값으로 바뀌게 된다. 복호화의 최종적인 값인 현재 프레임의 픽셀 값은 이전 프레임과의 차이 값으로 DCT-Domain Buffer에 저장된다. 일반적으로 비디오 스트림의 트랜스코딩에서 출력 비트율은 대개 입력 비트율보다 낮다. 동적 프레임율 조절을 위해 SW(Switch)가 사용되는데, 현재 매크로블록의 차이 값이 저장되는 DCT-Domain Buffer를 업데이트하기 위해 각 매크로블록의 부호화 모드 구분에 표 1과 같이 SW1과 SW2가 적용되고 프레임 제거에는 표 2와 같이 SW3이 사용된다.

그림 5의 비디오 트랜스코더는 Non-MC로 부호화된 매크로블록이 가진 차이 값을 픽셀 영역의 값으로 복호화하지 않고 DCT-Domain Buffer에 저장된 값과의 가산만을 수행하므로 전체적인 계산 복잡도를 낮추었다. 그러나 이 방법은 하나의 비디오 스트림에서 Non-MC 매크로블록의 비율이 높은 경우에만 좋은 성능을 낼 수 있고 대부분의 비디오 스트림에서 MC 매크로블록의 개수는 움직임 특성에 의존적이므로 트랜스코딩의 성능은 MC 매크로블록의 처리 방법에 좌우된다. 표 3에는 비디오 트랜스코더의 성능 평가에 자주 사용되는 비디오 스트림에서 Non-MC 모드 매크로블록의 평균 비율이 설명되어 있다. 이 때, 비디오 스트림에서 평균 MC 모

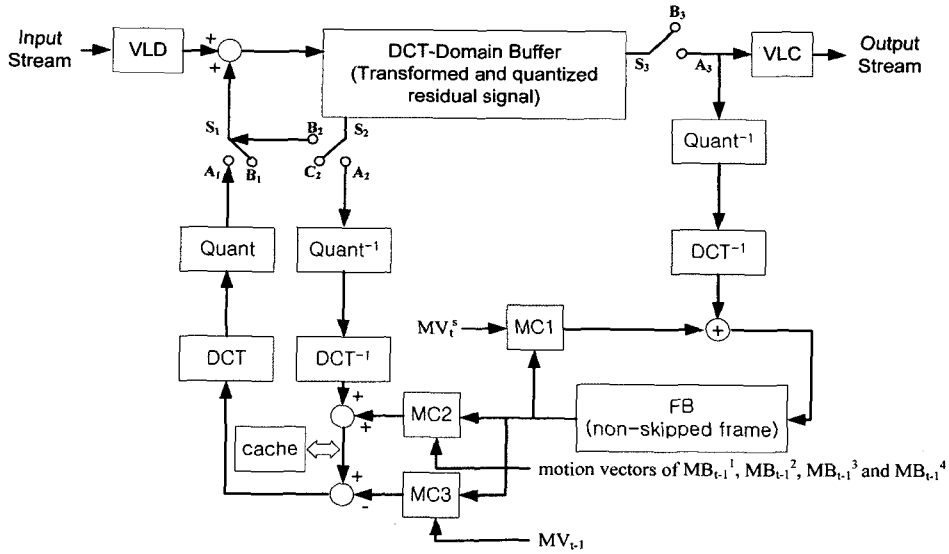


그림 5 매크로블록 모드에 적응적인 트랜스코더

표 1 부호화 모드와 SW1(S1)과 SW2(S2)의 위치

부호화 모드	S1	S2
MC	B1	B2
Non-MC	A1	A2

표 2 프레임 조절 방법과 SW3(S3)의 위치

프레임 조절 방법	S3
삭제되는 프레임	B3
남겨진 프레임	A3

표 3 다양한 비디오 스트림에서 Non-MC 모드 매크로블록의 평균 비율

Football	Carphone	Suzie	Tempete
18%	50%	56%	76%

드 매크로블록의 비율이 25% 이상이면 이전 프레임 전체를 복호화해야 하는 경우가 발생해서, 전체적인 계산 복잡도를 줄일 수 없다.

그리고, 그림 5의 트랜스코더는 MC 매크로블록을 픽셀 영역의 값으로 추출한 후 시간적 연관성을 보상하기 위해 움직임 벡터 선택 기법을 사용한다.

일반적으로 움직임 선택 기법은 그림 6과 같이 입력 스트림 구조에서 단순히 프레임 하나만 삭제되는 경우, MV'는 현재 프레임의 MV1과 삭제되는 프레임의 MV2를 단순히 더하여 재사용할 수 있다. 움직임 재예측을 하지 않으므로 계산 복잡도를 줄일 수 있지만 현재 매크로블록이 네 개의 인접 매크로블록 영역을 일부분씩 참조하고 있으므로, 하나의 매크로블록을 선택하는 기준이 중요하다. FDVS(Forward Dominant Vector Selection) 기법은 그림 7과 같이 현재의 매크로블록이 참조하고 있는 이전 프레임의 여러 매크로블록에 중첩되어 있을 때, 가장 넓은 공간적 영역을 차지하고 있는 매크로블록의 움직임 벡터를 선택한다. ADVS(Activity Dominant Vector Selection) 기법은 FDVS와 달리 중첩되는 매크로블록들이 가지는 활동상태 정보량을 이용하여 움직임 벡터를 선택한다. 중첩된 영역의 활동상태 정보량은 중첩된 매크로블록의 0이 아닌 DCT 계수의 개수를 계산하여 얻고, 그림 8은 ADVS에서의 계산 방법을 설명하고 있다.

FDVS는 계산량이 적고, ADVS는 움직임 예측이 정

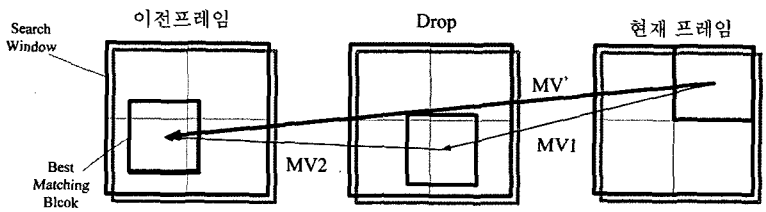


그림 6 움직임 벡터의 재구성

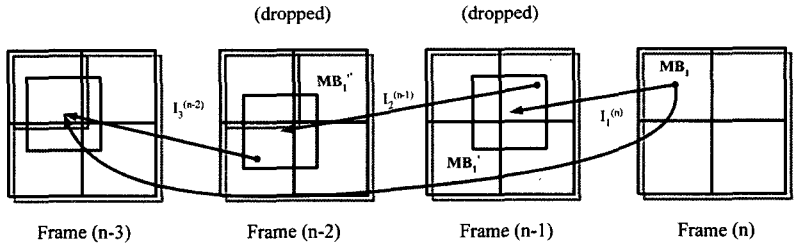
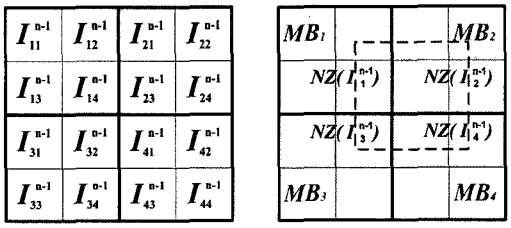


그림 7 FDVS 기법의 움직임 벡터 재구성

확한 장점을 가지고 있다. 그림 5에 나타난 [17]에서 제안한 트랜스코더는 계산량 감소를 위해 FDVS 기법을 사용하여 움직임 벡터를 선택하므로, 화질이 떨어지는 단점을 지닌다.



$$NZ(I_1^{n-1}) = NZ(I_{12}^{n-1}) + NZ(I_{14}^{n-1})$$

$$NZ(I_2^{n-1}) = NZ(I_{21}^{n-1}) + NZ(I_{22}^{n-1}) + NZ(I_{23}^{n-1}) + NZ(I_{24}^{n-1})$$

$$NZ(I_3^{n-1}) = NZ(I_{32}^{n-1})$$

$$NZ(I_4^{n-1}) = NZ(I_{41}^{n-1}) + NZ(I_{42}^{n-1})$$

$NZ(.)$: *number of nonzero quantized DCT coeffs*

그림 8 ADVS 기법의 움직임 벡터 재구성

3. 제안된 스트리밍 시스템 및 성능 분석

3.1 제안된 스트리밍 시스템

3.1.1 예측주기를 이용한 부호기

일반적으로 프레임율을 조절하는 트랜스코더에서 가장 큰 관심사는 시간적인 연관성이 있는 프레임 간의 예측 관계를 재구성하는 것이다. 그러므로 대부분의 트랜스코더에서는 삭제되는 프레임의 움직임 벡터를 재사용하여 움직임 예측에 활용하였다. 그러나 기존의 알고리즘은 트랜스코더 내부에만 한정되어 있으므로 계산 복잡도를 줄이는 데 한계가 있다.

본 논문에서는 기존의 서버 부호기에 "예측주기"를 이용하여 트랜스코더에서 제거될 프레임의 부가적인 처리를 줄일 수 있는 방법을 제안한다. 예측주기는 참조 프레임이 나타나는 프레임의 간격을 말한다. 예측주기를 스트림에 적용하면 연이은 프레임에 참조되지 않는 프레임이 있으므로 프레임율 조절과정에서 삭제 프레임에 소모되는 계산 복잡도를 줄일 수 있다. 예를 들어, 예측주기의 시작 프레임은 예측 구간에서 "예측주기-1"개의 프레임이 모두 참조하고 있으므로, 구간 내의 하나 혹은 여러 개의 프레임이 삭제되더라도 참조되는 프레임이 부호기 내 버퍼에 저장되어 반복적으로 재사용될 수 있다.

그림 9는 비디오 트랜스코더에 입력 스트림을 제공하는 서버의 부호기를 나타내고 있다. 예측주기에 따라 구성되는 비디오 스트림의 구조는 변경된다. 연이은 프레임에 참조된다면 참조 프레임으로써 스위치는 A의 위치에, 그렇지 않다면 B의 위치에 있게 된다. 예측주기는 동적인 프레임율 조절을 위해서 여러 가지 값을 가질 수 있는데, 그림 10에서는 예측주기가 2일 경우에 생성

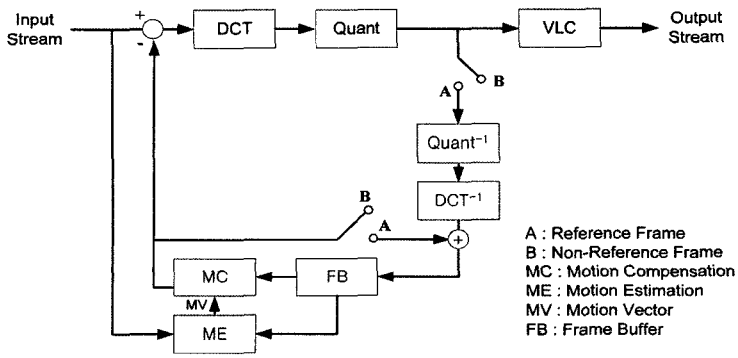


그림 9 서버의 부호기

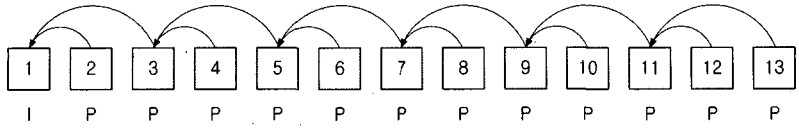


그림 10 예측주기 2가 적용된 비디오 스트림 구조

되는 프레임의 구조가 나타나 있다. 일반적으로 참조되는 프레임의 간격이 길어질수록 참조하는 프레임의 예측 오차는 커지기 때문에 이 점을 고려하여 본 논문에서는 프레임간의 예측주기를 2, 3, 그리고 4인 경우만 고려하였다.

3.1.2 동적으로 프레임율을 조절하는 트랜스코더

[17]에서 제안한 기법은 앞서 소개된 동적으로 프레임율 조절이 가능한 트랜스코더이고 그 성능은 입증된 바 있다. 각 매크로블록이 부호화된 모드, 즉 Non-MC 또는 MC에 따라 그 처리 과정을 달리하여 효율적인 트랜스코딩을 수행한다. 그러나 움직임 예측 방법으로 FDVS 기법을 그대로 사용하여 전체적인 계산 복잡도를 줄이려고 했으나 부정확한 움직임 벡터로 인해 화질의 열화를 발생시킨다.

이를 개선하기 위해 본 논문에서 제안하는 움직임 벡터 합성 기법을 앞서 설명된 동적인 프레임율 조절이 가능한 트랜스코더[17]에 적용하고 MMAT(Motion vector Mode Adaptive Transcoder)로 명명한 후 그림 11의 제안된 비디오 트랜스코더에 포함한다.

그림 12~그림 14에는 제안된 트랜스코더에서 동적 프레임율 조절의 세 가지 예가 설명되어 있다. 서버의 부호기에서 예측주기를 통해 비디오 시퀀스의 구조를 변경했다면, 트랜스코더에는 "삭제주기"가 적용되어 프레임율을 조절하게 된다. 삭제주기가 적용된 프레임은 NP 또는 P 그리고 NS 또는 S의 속성이 부여되는데, NP는 처리과정이 필요없는 프레임으로 Non-Proces-

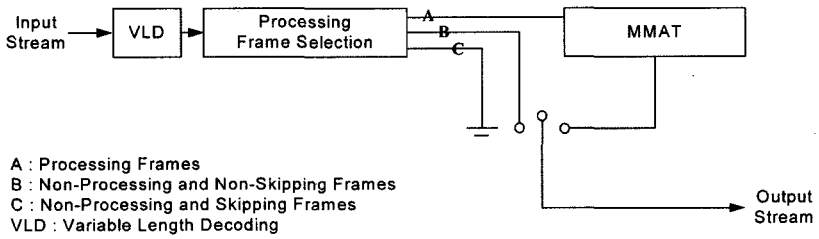
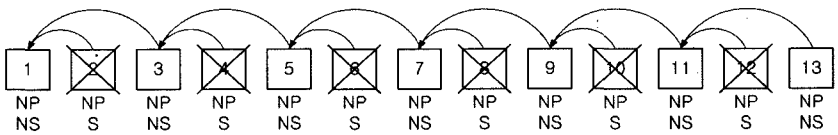
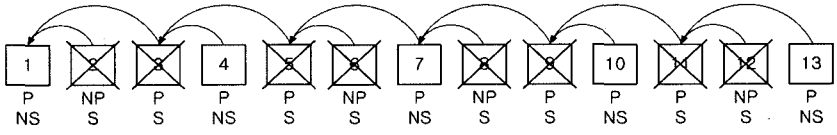


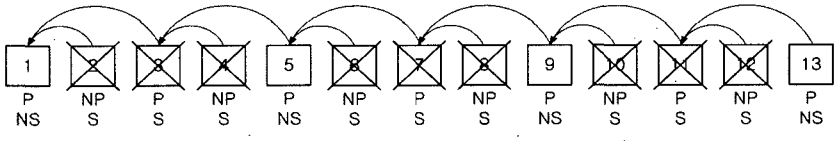
그림 11 트랜스코더 구조



(a) 2프레임에 하나의 프레임을 Skip할 경우

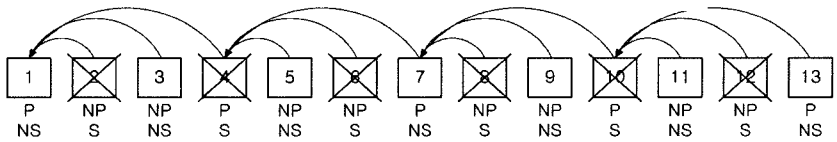


(b) 3프레임에 두 개의 프레임을 Skip할 경우

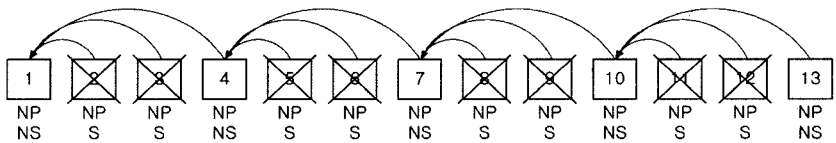


(c) 4프레임에 세 개의 프레임을 Skip할 경우

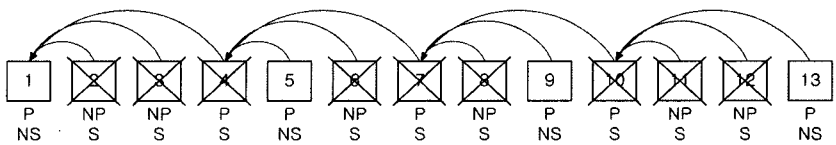
그림 12 예측주기가 2인 프레임을 제거할 경우



(a) 2프레임에 하나의 프레임을 Skip할 경우

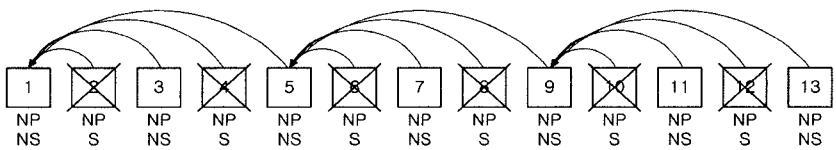


(b) 3프레임에 두 개의 프레임을 Skip할 경우

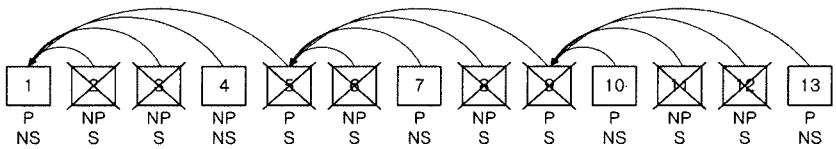


(c) 4프레임에 세 개의 프레임을 Skip할 경우

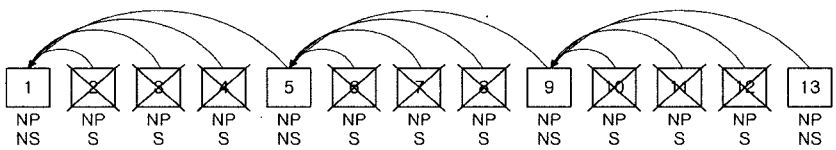
그림 13 예측주기가 3인 프레임을 제거할 경우



(a) 2프레임에 하나의 프레임을 Skip할 경우



(b) 3프레임에 두 개의 프레임을 Skip할 경우



(c) 4프레임에 세 개의 프레임을 Skip할 경우

그림 14 예측주기가 4인 프레임을 제거할 경우

sing, P는 처리과정을 필요로 하는 프레임으로 Processing, NS는 남겨지는 프레임으로 Non-Skipping, 그리고 S는 삭제되는 프레임으로 Skipping을 의미한다.

그림 12(a)에서 삭제되는 프레임이 출력될 프레임에 참조되지 않으므로 부가적인 처리 과정이 전혀 필요하지 않다. 그림 12(b)에서 2번 프레임은 삭제되어도 이후의 어떤 프레임에도 참조되지 않아서 부가적 처리를 필요로 하지 않지만, 3번 프레임은 출력될 4번 프레임에 의해 참조되고 있으므로 삭제되는 프레임이지만 처리해야 할 프레임이 된다. 그리고 1번 프레임은 4번 프레임이 3번 프레임을 통해서 움직임 정보 및 부호화 정보를 변환할 때 참조되므로 처리해야 할 프레임이다. 기존의 트랜스코더에서는 모든 프레임이 처리 대상이지만 예측주기가 적용된 비디오 스트림을 이용하면 트랜스코딩에서 처리해야 할 프레임을 많이 줄일 수 있다. 그림 13과 그림 14는 예측주기가 3과 4일 때의 경우를 설명하고 있다. 그림 12의 경우와 마찬가지로 설명할 수 있고, 각각의 경우에 따른 처리가 필요한 프레임의 개수는 표 4

표 4 예측주기 및 삭제주기에 따른 처리할 프레임율의 비교

삭제 주기 \ 예측 주기	2	3	4
2(2프레임당 하나의 프레임 삭제)	0%	33%	0%
3(3프레임당 두 개의 프레임 삭제)	67%	0%	42%
4(4프레임당 세 개의 프레임 삭제)	50%	50%	0%

와 같다. 표 4에는 예측주기와 삭제주기에 따라 처리할 프레임의 개수가 비교되어 있고 예측주기와 삭주기의 2, 3, 그리고 4의 최소공배수인 전체 12 프레임을 기준으로 계산되었다.

그림 15는 그림 12~그림 14에서 설명된 동적 프레임율 조절을 위한 Processing Frame Selection 알고리즘이다.

```
Procedure Processing_Frame_Selection()
Begin Procedure              // 요청된 프레임율에 따라 삭제주기 결정
nsp = 삭제 주기
pp = 예측 주기
If (pp == nsp * alpha)       // alpha : 양의 정수
  Exit Procedure             // Bypass의 경우
End If
lcm = LCM(pp, nsp)           // pp와 nsp의 최소공배수
last_pp_skip=1               // 마지막 예측주기 프레임의 삭제 여부
                             // 1: 삭제된 프레임, 0: 남은 프레임
k = 1                        // 프레임 번호
For ( k <= lcm )
  If((k-1) % pp == 0)
    Processing Frame         // 처리할 프레임
  Else
    If(pp <nsp)              // 삭제주기 > 예측주기
      If((k-1) % nsp == 0)
        Processing Frame // 처리할 프레임
      End If
    Else                     // 삭제주기 < 예측주기
      If(last_pp_skip == 1 && (k-1) % nsp == 0)
        Processing Frame // 처리할 프레임
      End If
    End If
  End If
  If((k-1) % pp==0 && (k-1) % nsp==0) // 마지막 예측주기 프레임의 삭제 여부
    last_pp_skip = 0
  End If
  k++;                       // 다음 프레임
End For
End Procedure
```

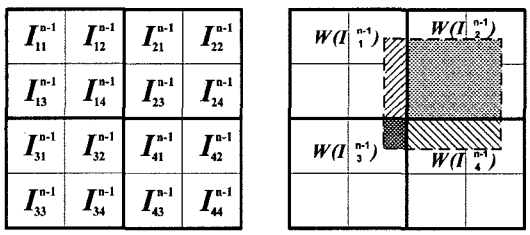
그림 15 Processing Frame Selection 알고리즘

• 움직임 벡터 합성 기법

ADVS 기법은 FDVS와 마찬가지로 현재의 매크로블록이 두 개 이상의 이전 매크로블록에 중첩되어 있고 각 매크로블록의 활동상태 정보량이 모두 같다면 제시된 기준대로 움직임 벡터를 선택하기 어렵다. 그리고 매크로블록 단위가 아닌 8×8 블록의 0이 아닌 DCT 계수의 개수를 기준으로 활동상태 정보량을 측정하는데, 이전 매크로블록에서 중첩된 일부 블록의 DCT 에너지만으로 네 개 블록 모두의 DCT 에너지인 것처럼 취급한다. 즉, 중첩되지 않은 나머지 블록에 더 많은 활동상태 정보량이 포함될 수도 있으므로 일부의 측정된 값으로 전체의 활동상태 정보량을 결정하는 것은 무리가 있고 이는 화질 저하를 초래할 수 있다.

본 논문에서는 가장 유력한 매크로블록을 선택하는 것이 아니라 새로운 가중치를 기반으로 중첩된 매크로블록의 움직임 벡터들을 합성하는 기법을 제안한다. 새로운 가중치는 중첩 영역의 넓이와 활동상태 정보 모두를 고려하여 계산되고 중첩된 각 매크로블록의 움직임 벡터에 적용된 후에 합성되어 하나의 새로운 움직임 벡터가 된다. 새로운 가중치는 중첩된 영역의 면적과 활동상태 정보량이다. 중첩된 영역의 넓이는 픽셀 단위로 계산하고 활동상태 정보량은 ADVS 기법에서 적용된 8×8 블록 단위의 정보가 아닌 움직임 벡터의 단위인 매크로블록 단위로 측정되어 각 매크로블록 전체의 정보를 반영한다.

그림 16에서는 네 개의 블록으로 이루어진 인접한 네 개의 매크로블록과 각각에 해당하는 중첩 영역의 가중치가 나타나 있다. 각 매크로블록에 중첩된 영역의 가중치는 아래의 과정을 통해 계산할 수 있다.



(a) 네 개의 인접한 매크로블록 (b) 중첩영역의 가중치

그림 16 네 개의 인접한 매크로블록에서 가중치 계산

Step 1. 중첩된 각 매크로블록의 가중치 계산

$$Wi = (NZ(Ri) \times OAi\ ) / \sum_{i=1}^{4} (NZ(Ri) \times OAi)$$

ADVS 기법과 달리 매크로블록 단위로 0이 아닌 DCT 계수의 개수와 픽셀 단위의 중첩된 영역을 계산한다. 항목 $NZ(Ri)$ 와 OAi 는 각각 매크로블록의 활동상태 정보량과 중첩된 영역의 크기이다. 각 매크로블록의 움직임 벡터에 적용되는 가중치는 중첩된 모든 움직임 벡터의 가중치의 합에 대한 그 움직임 벡터의 가중치의 비율이다.

Step 2. 현재 매크로블록의 움직임 벡터 합성

$$\text{Composed MV} = \sum_{i=1}^{4} MVi \quad * Wi$$

Step 1을 통해서 얻어진 각 매크로블록의 가중치와

해당 매크로블록의 움직임 벡터의 곱을 통해서 새로운 움직임 벡터를 합성한다.

3.1.3 트랜스코딩 여부를 판별하는 클라이언트의 복호기

대부분의 사람들이 사용하고 있는 Microsoft™의 Windows에 기본적으로 설치되어 있는 Window Media Player[3], 스트리밍 서비스로 유명한 RealNetworks™의 RealPlayer[5], 그리고 Apple™의 QuickTime[4]은 전용 재생기를 제공하고 있다. 이와 같은 전용 재생기는 간단한 플러그인의 설치만으로 비디오 스트리밍 서비스에 많이 사용된다.

본 논문에서 제안한 서버의 부호기로 변경된 비디오 스트림은 트랜스코딩을 거치지 않으면 일반 클라이언트에서 제대로 재생되지 않을 수도 있다. 그래서 그림 17과 같이 하나의 스위치를 추가하여 트랜스코딩의 여부에 따라 올바로 재생될 수 있도록 제안하였다. 일반적인 복호기에 비해 큰 처리 비용을 필요로 하지 않기 때문에 열악한 환경의 클라이언트에도 어렵지 않게 사용될 수 있다.

3.2 성능 분석

본 절에서는 제안된 스트리밍 시스템의 성능 평가를 위해 여러 가지 인자를 적용하여 실험하였다. 먼저 일반 부호기-[17]의 트랜스코더-일반 복호기와 제안 부호기-제안 트랜스코더-제안 복호기 스트리밍 시스템 구조에서 각각의 트랜스코더 내부의 처리시간과 출력 비디오 스트림의 화질로 성능을 평가한다. 또한, 제안된 부호기의 특성이 전체 시스템에 미치는 영향을 조사하기 위해서 일반 부호기와 제안 부호기에서의 처리시간과 출력 스트림의 화질을 비교하였다.

실험에 사용된 비디오 스트림은 QCIF 포맷의 H.263 [23] 파일이고 "IPPP..."의 구조를 가진다. 성능 평가를 위해 표준 비디오 파일 이외에 실제 서비스에 사용되는 뮤직비디오도 실험에 사용되었다. Suzie, Football, 뮤직비디오 1, 그리고 뮤직비디오 2는 전체 스트림에서 각각 44%, 82%, 61%, 그리고 52%의 0이 아닌 움직임 벡터를 포함하고 있다. 실험에 사용된 부호기 및 트랜스코더는 FFmpeg[24]의 H.263 코덱을 수정하여 구현하였다. 표 5와 표 6에는 각각 입력 비디오 스트림과 실험 환경이 나타나 있다.

그림 18~그림 21은 [17]에 제안된 트랜스코더와 본 논문에서 제안한 트랜스코더의 예측주기가 2, 3, 4일 때, 삭제주기가 2인 경우의 움직임이 많은 스트림에 대한 계산 복잡도(㎲)와 출력화질(dB)을 나타내고 있다.

표 7에는 네 개의 비디오 스트림에서 기존의 트랜스코더와 제안된 트랜스코더의 평균 PSNR과 처리시간이 나타나 있다. 그림 22~그림 25은 예측주기별 출력 스트림의 특정 프레임별 화질을 비교하여 나타내고 있다. 삭

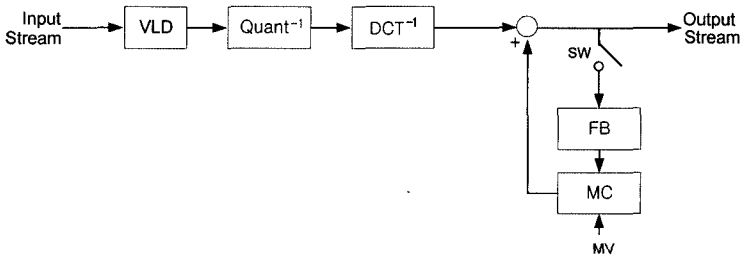


그림 17 클라이언트 복호기

표 5 실험 환경

시스템 환경	CPU & 메모리	Pentium IV 1.2 Ghz CPU, 256M
	운영체제	Redhat Linux 9.0
비디오 타입	H.263	"IPPP.." 시퀀스
비디오 코덱	FFmpeg	Version 0.4.8
출력 프레임율	15, 10, 7.5	삭제 주기를 2~4로 조절
출력 양자화 인수	5	

표 6 입력 비디오 스트림

시퀀스	양자화 파라미터	Video Format	프레임율 (fps)	프레임 개수
Football	5	QCIF	30	257
Suzie				149
Music Video-1				300
Music Video-2				300

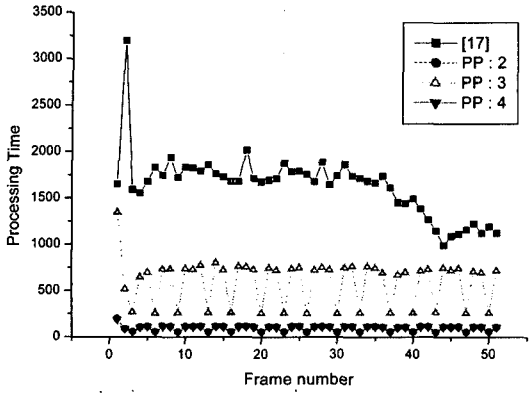


그림 18 football 스트림에서 예측주기별 계산복잡도

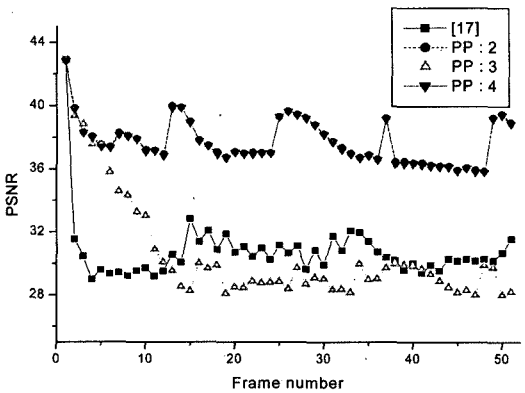


그림 20 football 스트림에서 예측주기별 화질

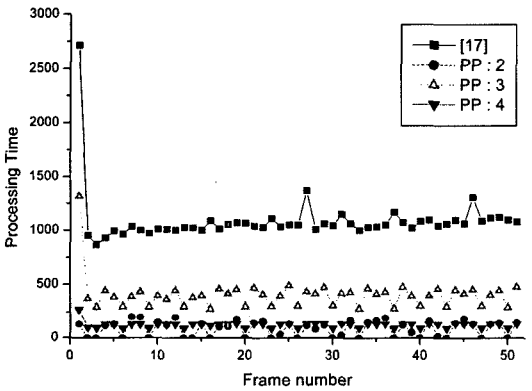


그림 19 뮤직비디오1 스트림에서 예측주기별 계산복잡도

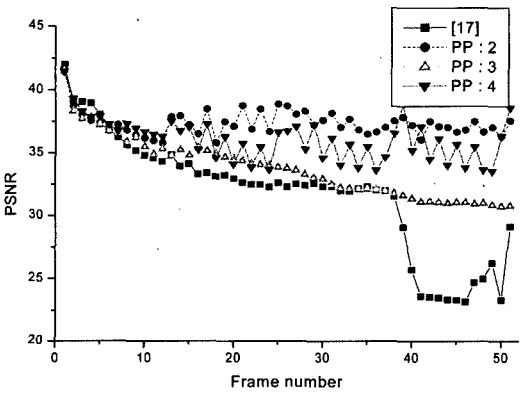


그림 21 뮤직비디오1 스트림에서 예측주기별 화질

표 7 트랜스코더의 예측주기와 삭제주기별 성능 비교

비교항목			football	suzie	MV-1	MV-2
기존트랜스코더 (삭제주기:2)		평균 PSNR	30.72	38.21	31.75	33.06
		처리시간(μs)	198985	71786	182946	182629
예측주기:2	삭제주기:2	평균 PSNR	37.88	39.66	37.30	38.56
		처리시간(μs)	11481	4735	16873	18155
	삭제주기:3	평균 PSNR	30.18	37.23	30.96	33.28
		처리시간(μs)	131149	46958	122765	124872
	삭제주기:4	평균 PSNR	30.01	36.53	30.17	32.22
		처리시간(μs)	117645	38997	92813	93447
예측주기:3	삭제주기:2	평균 PSNR	30.94	36.29	33.64	34.89
		처리시간(μs)	67782	25182	62735	61234
	삭제주기:3	평균 PSNR	37.80	39.10	36.54	37.82
		처리시간(μs)	13039	4745	16848	17595
	삭제주기:4	평균 PSNR	29.83	36.08	30.09	31.74
		처리시간(μs)	101618	37217	93637	93796
예측주기:4	삭제주기:2	평균 PSNR	37.72	39.02	36.19	37.34
		처리시간(μs)	11223	4958	18633	19387
	삭제주기:3	평균 PSNR	29.68	34.38	30.66	32.2
		처리시간(μs)	83198	31198	78693	78315
	삭제주기:4	평균 PSNR	37.72	39.02	36.19	37.34
		처리시간(μs)	11739	4435	15792	11743

(a) (b) (c) (d)

그림 22 suzie 스트림(#72)에서 예측주기별 화질 비교

(a) (b) (c) (d)

그림 23 suzie 스트림(#72)에서 예측주기별 화질 비교

(a) (b) (c) (d)

그림 24 뮤직비디오1 스트림(#252)에서 예측주기별 화질 비교

(a) (b) (c) (d)

그림 25 뮤직비디오2 스트림(#264)에서 예측주기별 화질 비교

제주기가 2일 때 기존의 트랜스코더를 거친 프레임과 제안된 트랜스코더를 거친 프레임이 나타나 있다. 각 그림별 (a)는 기존의 트랜스코더에서의 프레임을 나타내고, (b)~(d)는 제안된 트랜스코더에서 예측주기가 2, 3, 4일 때의 프레임이다.

앞서 제시된 표와 그림을 통해서 제안된 트랜스코더는 [17]에서 제안된 트랜스코더에 비해 비슷하거나 조금 더 나은 PSNR을 보인다. 또한 처리시간은 예측주기가 삭제주기의 정수배인 특별한 경우를 제외하더라도 전체적으로 절반 가량의 시간을 필요로 한다. 그러므로 처리시간과PSNR에서 기존 기법보다 우수한 성능을 보임을 확인할 수 있다. 또한, 기존 시스템과 제안된 시스템에서 서버측 부호기 특성, 즉 부호기의 예측주기에 따른 전체 시스템의 성능 변화 가능성 조사를 위하여 표 8에 각각의 스트림에서 기존의 부호기와 제안된 부호기의 평균 PSNR과 처리시간 결과를 비교하였다. 실험 결과에서 알 수 있듯이, 제안부호기의 예측주기가 커질수록 PSNR이 작아지지만 처리시간도 줄어든다는 것을 확인할 수 있다. 따라서, 스트리밍 시스템의 성능은 트랜스코더에 좌우된다고 볼 수 있다.

표 8 부호기의 예측주기별 성능 비교

비교항목		football	suzie	Music Video-1	Music Video-2
일반 부호기	평균 PSNR	38.63	39.91	38.77	38.84
	처리시간(μs)	775348	594388	897886	940065
제안 부호기 (예측주기:2)	평균 PSNR	37.88	39.66	37.30	38.56
	처리시간(μs)	697813	534949	808097	846059
제안 부호기 (예측주기:3)	평균 PSNR	37.80	39.10	36.54	37.82
	처리시간(μs)	682306	523061	790140	827257
제안 부호기 (예측주기:4)	평균 PSNR	37.72	39.02	36.19	37.34
	처리시간(μs)	666799	511174	772182	808456

4. 결론 및 향후과제

본 논문에서는 동적으로 프레임율 조절이 가능한 스트리밍 시스템을 제안 하였다. 프레임율을 조절하는 트랜스코딩에서 계산 복잡도를 최소화하기 위해서 서버의 부호기에서 예측주기를 이용하였다. 트랜스코더는 기존에 제시된 우수한 트랜스코더[17]에 개선된 움직임 예측을 위해 움직임 합성 기법을 적용하였으며, 삭제주기에 따라 기존의 트랜스코더에 비해 처리시간을 줄이도록 구성하였다.

트랜스코더에 입력되는 비디오 스트림은 예측주기가 적용되어 프레임율 조절이 용이한 구조를 가지므로 프레임율 조절 시에 기존의 트랜스코더 구조보다 평균 60%이상의 처리 시간을 줄였다. 또한, 움직임 벡터 합성 기법을 적용하여 기존의 움직임 벡터 재사용 기법보다 대략 6% 정도로 향상된 화질의 비디오 스트림을 출력하였고 비디오 스트림의 움직임이 많을수록 향상의 폭도 컸다.

본 논문에서 제안하는 시스템은 단방향 예측의 비디오 스트림에만 적용 가능하다. 따라서, 향후에는 양방향 예측이 있는 비디오 스트림에서도 프레임율을 효율적으로 조절할 수 있는 연구가 필요하다.

참 고 문 헌

[1] M. S. Chen and D. D. Kandlur, "Downloading and stream conversion: Supporting interactive playout of videos in a client station," in Proc. 2nd Int. IEEE Conf. Multimedia Computing and Systems, 1995, pp. 7380.

[2] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-ondemand," IEEE Multimedia, vol. 13, pp. 1424, Aug. 1994.

[3] MicrosoftWindows Media, Microsoft Corporation Inc. [Online]. Available: http://www.microsoft.com/windows/windowsmedia/

[4] Apple QuickTime Player, Apple Corporation Inc. [Online]. Available: http://www.apple.com/quictime/

[5] Real Networks RealPlayer [Online]. Available: http://www.real.com/

[6] Relay Networks ReplayTV [Online]. Available: http://www.replay.com/

[7] TiVo Inc [Online]. Available: http://www.tivo.com/

[8] A. Vetro and C. Christopoulos and H. Sun, "Video Transcoding Architectures and Techniques: An Overview," IEEE signal Processing Magazine, ISSN: 1053-5888, Vol. 20, Issue 2, pp. 18-29, March 2003.

[9] T. Warabino, S. Ota, D. Morikawa, M. Ohashi, H. Nakamura, H. Iwashita and F. Watanabe, "Video Transcoding Proxy for 3Gwireless Mobile Internet Access," IEEECommunications Magazine, 38, (10), pp. 66-71, Oct. 2000.

[10] N. Bjork and C. Christopoulos, "Video Transcoding for Universal multimedia Access," ACM Workshop on Multimedia Standards, Interoperability and Practice (MM2000 Workshop), Proceedings of ACM Multimedia 2000, pp. 75-79, November 4, 2000.

[11] T. shanableh and M. Ghanbari, "Heterogeneous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats," IEEE Trans. Multimedia, vol. 2, no. 2, pp. 101-110, June 2000.

[12] Y. Nakajima and M. Sugano, "MPEG bit rate and format conversions for heteroheneous network/storage applications," IEICE Trans. Electron., E85-C, (3), pp. 492-503, Mar. 2002.

[13] M. Emad Modirzadeh, "Bit-Rate Reduction of MPEG Compressed Video," IEEE Canadian Conference on Electrical and Computer Engineering, 2002.

[14] H. Sorial, W. Lynch, A. Vincent, "Selective Requantization for Transcoding of MPEG Compressed Video," IEEE International Conference on Multimedia and Expo (ICME 2000), NY, USA, 30 July - 2 August 2000.

[15] J. Youn, M.T. Sun, and J. Xin, "Video Transcoder Architectures for Bit Rate Scaling of H.263 Bit Streams," Proceedings of ACM Multimedia'99, pp. 243-250, Nov., 1999.

[16] H.Sun, W. Kwok, and J. W. Zdepski, "Architecture

for MPEG compressed bitstream scaling," IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 191-199. Apr. 1996.
[17] K. T. Fung, Y. L. Chan and W. C. Siu, "New architecture for dynamic frame-skipping transcoder," in Proc. IEEE Workshop Multimedia signal Processing, Redondo Beach, CA, pp. 616-621, Dec. 1998.
[18] M. J. Chen, M. C. Chu and C. W. Pan, "Efficient motion estimation algorithm for reduced frame-rate video transcoder," IEEE Trans. Circuits syst. Video Technol., vol. 12, pp. 269-275, Apr. 2002.
[19] J. Youn, M. T. Sun and C. W. Lin, "Motion vector refinement for high performance transcoding," IEEE Trans. Multimedia, vol. 1, pp. 30-40, Mar. 1999.
[20] B. Shen, I. K. Sethi and B. Vasudev, "Adaptive motion vector resampling for compressed video downscaling," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 929-936, Sept. 1999.
[21] V. Bhaskaran and K. Konstantinides. Image and Video Compression Standards. Kluwer Adademic Publishers, 1994.
[22] N. Bjrk and C. Christopoulos, "Transcoder architectures for video coding," IEEE Trans. Consumer Electron., vol. 44, pp. 88-98, Feb.1998.
[23] "Video coding for low bitrate communication," International Telecommunications Union, Geneva, Switzerland, ITU-T Recommendation H.263, 1998.
[24] FFmpeg multimedia system. [Online]. Available: http://www.ffmpeg.org

김 성 민

2001년 부산대학교 전자계산학과 졸업(학사). 2003년 부산대학교 전자계산학과 졸업(이학석사). 2003년~현재 부산대학교 컴퓨터공학과 박사과정. 관심분야는 트랜스코딩, 멀티미디어 스트리밍, 유비쿼터스 컴퓨팅

김 현 희

2003년 신라대학교 멀티미디어공학과 졸업(학사). 2005년 부산대학교 컴퓨터공학과 졸업(공학석사). 2005년~현재 동양종합금융증권(주) 컨텐츠팀. 관심분야는 트랜스코딩, 유비쿼터스 컴퓨팅, 멀티미디어 스트리밍

박 시 용

1997년 경성대학교 전자계산학과 졸업(학사). 2001년 부산대학교 대학원 멀티미디어과 졸업(이학 석사). 2005년 부산대학교 대학원 전자계산학과 졸업(이학박사). 2006년~현재 대전대학교 교육 개발 센터 전임강사. 관심분야는 멀티미디어, 모바일 네트워크, 인터넷 QoS, 유비쿼터스 컴퓨팅

정 기 동

1973년 서울대학교 졸업(학사). 1975년 서울대학교 대학원 졸업(석사). 1986년 서울대학교 대학원 계산통계학과 졸업(이학박사). 1990년~1991년 MIT, South Carolina 대학 교환 교수. 1995년~1997년 부산대학교 전자계산소 소장. 1978년~현재 부산대학교 전자계산학과 교수. 관심분야는 멀티미디어 시스템, 멀티미디어 통신, 병렬처리